BraiNet: Group 2 Project Report

Authors:
Aditi Sonik, Chaynika Saikia, Niharika Dutta, Sarthak Khanna

Abstract— **Security and user authentication is becoming a thing of utmost importance in today's world. Every user wants their information to be secure and not get into the wrong hands. One way to do this is to use EEG data, which are the brain signal of a person. These signals can be used to authenticate the user and are almost impossible to replicate or forge. This project aims to develop an Android application which uses EEG data (or brain signals) as a method of authentication and identification. The project uses machine learning techniques to authenticate the user and is contextually aware, to avoid using excessive resources.**

Keywords: Mobile Computing, Fast Fourier Transform, Security, Authentication, Support Vector Machines, EEG, User Identification, Offloading Algorithm.

I. INTRODUCTION

The objective of the project is to enhance the security of user data by making an authentication system, which uses biometric information to validate the user. The project specifications required to develop an Android application that uses user's brain signals as a way of authentication. For implementation, the first step is to build a database which contains Brain signal data for a sample of users. The brain signal data was extracted from the EEG Motor Movement/Imagery Dataset provided by PhysioNet [1], which is collected using 64 sensors for each brain signal or data point. The dataset provided by them was huge in size. Due to application constraints we applied some data cleaning techniques and used a small size of dataset for the Android application. Since the actual brain signals cannot be collected from the user in real-time (since sensors are not provided as a part of this project), the brain signals dataframe has been split into train and test data for each user. According to the specifications the application requires user to login through their brain signals, a mechanism was needed to compare the obtained user signals or the test data to the training data. The computations, required to be done for comparing the user's brain signal to the one present in database for login, were too expensive to be performed on hand-held device. Compute intensive technologies like cloud and fog servers were used to perform complex computations and return valid response back to the user.

In order to compare the test data to the training data, a Support Vector Machine model is used, to authenticate the user at the time of logging in.

Making an application context aware is also important so that it can make decisions about which server to communicate with for faster computation. This has been implemented by developing an offloading algorithm which helps the application to choose a server on the fly. The offloading algorithm takes network delays and current battery status of the hand-held device and accordingly decides which server to contact. Detailed implementation of the algorithm is given in section [II (H)].

The remainder of this paper is organized in the following manner. Section II of the paper talks about the implementations of the User Interface, user interaction with the application, the data cleaning phase, and setting up the cloud and fog server. The results of the application can be viewed in Section III of the paper. The major challenges encountered during the application development are dealt in section four. Section V talks about learning and outcomes of the project. There is ample scope of improvement in several parts of the application and hence future work is discussed in Section VI of the

paper. It is finally followed by a conclusion in Section VII.

II. IMPLEMENTATION

The implementation details have been explained in the following sections.

A. Frontend

The client application enables the user to register and login using their EEG signals. Figure 1 shows the initial activity (home screen) of the app, it consists of a text entry to enter the username, a Register button, an Unlock Phone button and a textview to show successful authentication. The user enters their username in the text field and presses the Register button. This populates the database with the username and also randomly selects an EEG data file and associates the username with the training signal of that particular EEG file. This is the dataset that is used to train the SVM model. Then these username and the EEG files are sent to both the cloud and the fog server for initial registration. Once the user is registered, they can unlock the phone by clicking the Unlock phone button. This calls the offloading algorithm [II (H)] which decides which server to use (fog or cloud). Once the server is decided, the Check Signature API [II (B)] is invoked on the selected server, this uses the trained SVM model (saved in the fog and cloud servers) and checks the user sent EEG test signals against the model. If it is a match, it returns an HTTP response and authenticates the user, otherwise an unauthorised user toast is displayed. Figure 2 shows successful authentication using the fog server and Figure 3 shows successful login using the cloud server.

B. Client - Server API's

There are two APIs implemented at both the fog and the cloud servers:

UploadToServer: This API is responsible for uploading the train and test EEG signals from the client to both servers.

CheckSignature: This API calls the comparator code that predicts the test EEG signal dataset by testing against the SVM model, and returns appropriate JSON response depending on whether it is successful or not to authenticate the user.

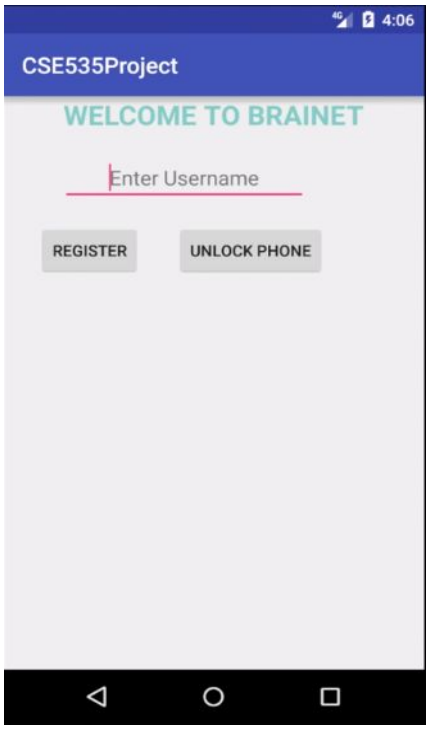

Figure 1. Home screen

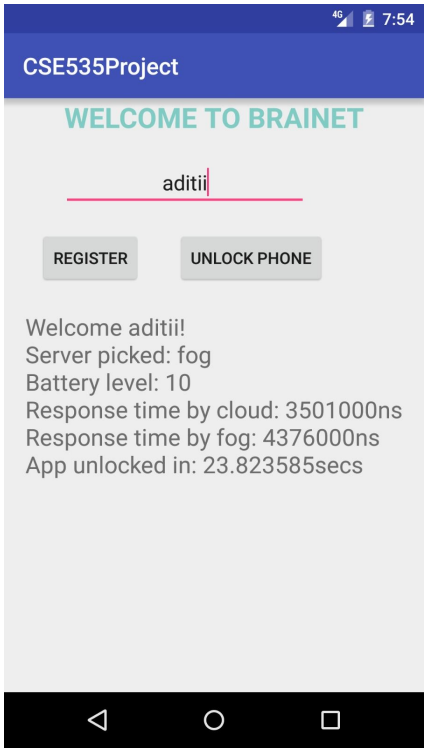

Figure 2. Successful login using FOG

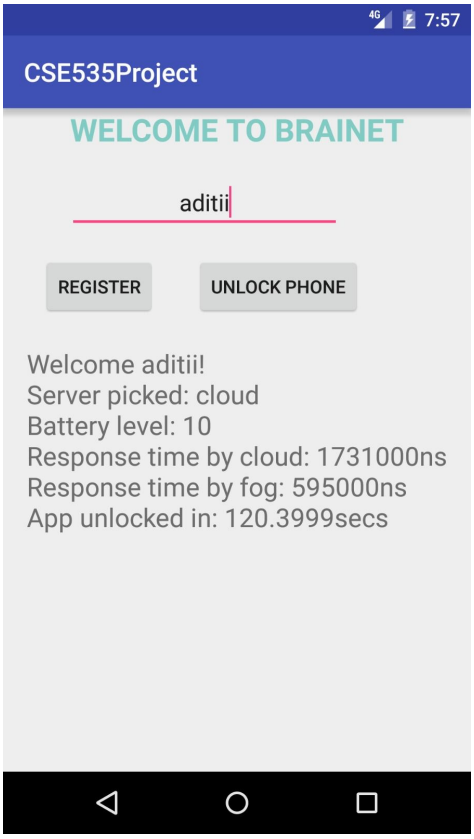

Figure 3. Successful login using Cloud

C. *Fog Server*

The fog server was run on MacOS Sierra having a 2.7 GHz Intel Core i5 processor and 8 GB RAM. The phone and the fog server were on the same network.

D. *Cloud Server*

The cloud server was running as a Digital Ocean droplet with very limited resources. The server configuration was 1GB RAM, 30 GB disk space and a single core 1.79 GHz CPU. LAMP stack was installed on Ubuntu 16.04. The server was based in New York.

E. *Data Preprocessing*

Due to the application requirements we were required to have actual brain signals for development. One option was to collect the brain signals through NeuroSky Mindwave EEG headset sensor but there were few shortcomings with this approach. As emotions played a vital role in the type of brain signals, there was a high probability of the brain signal data for same person with different emotions to vary a lot. Also the accuracy of signals with Mindwave headset was low as it sometimes captured the muscle movement impulse instead of an actual brain signal. Hence, the dataset used for this project was collected from PhysioNet, namely the EEG Motor Movement/Imagery Dataset [1]. They recorded the Brain signals by monitoring 64-channel EEG using the BCI 2000 system [2]. Each subject was required to perform 14 different tasks, like open eyes,close eyes, etc, and their Brain signals through these 64 points were recorded. The dataset consists of data for 104 subjects, which makes it enormous in size. Due to space constraints and processing limitations we built a small dataset of 10 subjects and used only 1 EEG sensor signals instead of 64. The resulting dataset was approximately of size 5MB and in this manner overcame the challenge of building database for our application. This dataset was later subdivided into train and test data.

F. *FFT and Feature extraction*

FFT transforms the signal dataset to the frequency domain which makes calculation and training for the SVM model easier to do. We also extracted the features by taking only data corresponding the first sensor.

G. *SVM Classifier*

An SVM classifier has been used as the classification model [3]. The model is trained on the training data, since the corresponding user of each brain signal is provided at the time of registration (class labels are given). This trained model is used to predict the user each time he tries to login (which is the test data). The trained model was used on all 64 sensors for each data point, which meant the SVM classifier took excessive time to train and predict. The average response time to indicate successful or unsuccessful authentication took around 2 minutes, which was not ideal. So dataset was changed to include only sensor 1 information. In order to reduce the time, the changed dataset was utilized.

H. *Offloading Algorithm*

The offloading algorithm implemented by the paper was inspired by the work of Gu, Xiaohui, et al [4] . While developing the offloading algorithm, inspiration was taken from [5], [6] and [7]. The above

mentioned offloading algorithm takes network delay on sending a request to the server and device's battery status into consideration for deciding which server to use for authentication. The network delay is measured by first pinging both the servers and storing the round trip time. The amount of battery used in contacting the servers is also noted. If the battery level of the device is less than 25% then the server which consumes less battery in sending and receiving the packet is selected by the algorithm.

I. Task Execution

#	Task	Contributor
1	Creating the UI	Aditi
2	Getting brain signal information from SD card.	Sarthak
3	Testing the UI	Chaynika
4	Remote server setup and configuration	Sarthak
5	Database setup and connection/configuration	Niharika
6	Developing and training the SVM model and writing comparator to test authentication	Niharika, Chaynika
7	Designing an UI for user query	Sarthak
8	Testing database configuration and operations	Sarthak
9	Fog server setup in the same network as smartphone	Aditi, chaynika
10	Implement recognition algorithm in fog server	Chaynika
11	Sense smartphone status including battery level and network delay	Sarthak
12	Run an algorithm in the smartphone to determine whether to use fog or remote server	Chaynika
13	End to end implementation of the recommendation of the algorithm	Niharika
14	Run application on end to end system	Niharika
15	Report accuracy of detection	Aditi
16	Recording Execution times for both fog and cloud server	Aditi
17	Record power consumption of the smartphone when using fog server or cloud server	Aditi
18	Compare fog server only v.s. remote server only, v.s. adaptive offloading techniques with respect to execution time and power.	Niharika

Table 1. Task list

III. RESULTS

A number of test cases were run to study and understand our application from the perspective of parameters like round trip time for getting response, server being used, response from fog or cloud server, and battery level. These values are captured in the table below.

Accuracy of detection	93%

Table 2: Fog vs. Cloud vs Adaptive offloading

Parameters	Fog	Cloud Server
Execution time	23.82 s	120.39s
Power consumption	17mAh	28mAh

Table 3: Fog vs. Cloud vs Adaptive offloading

Parameters	Fog	Cloud Server	Adaptive offloading Algorithm
Execution time	23.82	120.39	23.82
Power	17 mAh	28mAh	17mAh

Table 4: Fog vs. Cloud vs Adaptive offloading

IV. CHALLENGES

A number of challenges were faced while developing this application. First, it was realized that training and testing on an SVM model using the entire dataset of brain signal with 64 sensors was proving to take too much time, which resulted in a big response time for the application. So a decision was taken to prune our dataset to contain data from only the first sensor, which resulted in slightly reduced accuracy, but much faster execution time.

The comparator code provided to us by the IMPACT lab at ASU, was proving to be a bit of a bottleneck in terms of integrating it with the application, so a new algorithm was developed to compare the train and test signals, i.e the SVM classifier.

V. LESSONS LEARNED

This project was an amazing learning experience which taught us the intricacies of robust application development in Android. We also learned how Machine learning works in the environment of an application, and understood the fundamentals and working of a Support Vector Machine. We also understood the importance and application of fog and cloud servers and the parameters that go into developing a good offloading algorithm.

VI. FUTURE WORK

In future release of this application, some areas of improvement include fine tuning the SVM model and optimizing the battery consumption of the app. The implemented offloading algorithm can also be improved by adding more parameters to it. The performance of the application can also be improved by configuring the cloud server on powerful hardware. Another interesting research paper we came across on offloading was the work of Rachuri, Kiran K., et al, that can be incorporated in the application in the future.

VII. CONCLUSION

This application was developed with a purpose to provide an authentication system with a high bar for security, using brain or EEG signals of the user. The impact of developing an application that relies on both cloud and fog server was studied and it was observed that this resulted in a more robust and reliable application with higher fault tolerance and better resource utilization.

REFERENCES

[1] https://www.physionet.org/pn4/eegmmidb/

[2] http://www.bci2000.org/

[3]https://en.wikipedia.org/wiki/Support_vector_machine

[4] Gu, Xiaohui, et al. "Adaptive offloading inference for delivering applications in pervasive computing environments." *Pervasive Computing and Communications, 2003.(PerCom 2003). Proceedings of the First IEEE International Conference on*. IEEE, 2003

[5]Rachuri, Kiran K., et al. "METIS: Exploring mobile phone sensing offloading for efficiently supporting social sensing applications." *Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on*. IEEE, 2013.

[6] Metri, Grace, Weisong Shi, and Monica Brockmeyer. "Energy-efficiency comparison of mobile platforms and applications: A quantitative approach." *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM, 2015.

[7] Schilit, Bill, Norman Adams, and Roy Want. "Context-aware computing applications." *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*. IEEE, 1994.